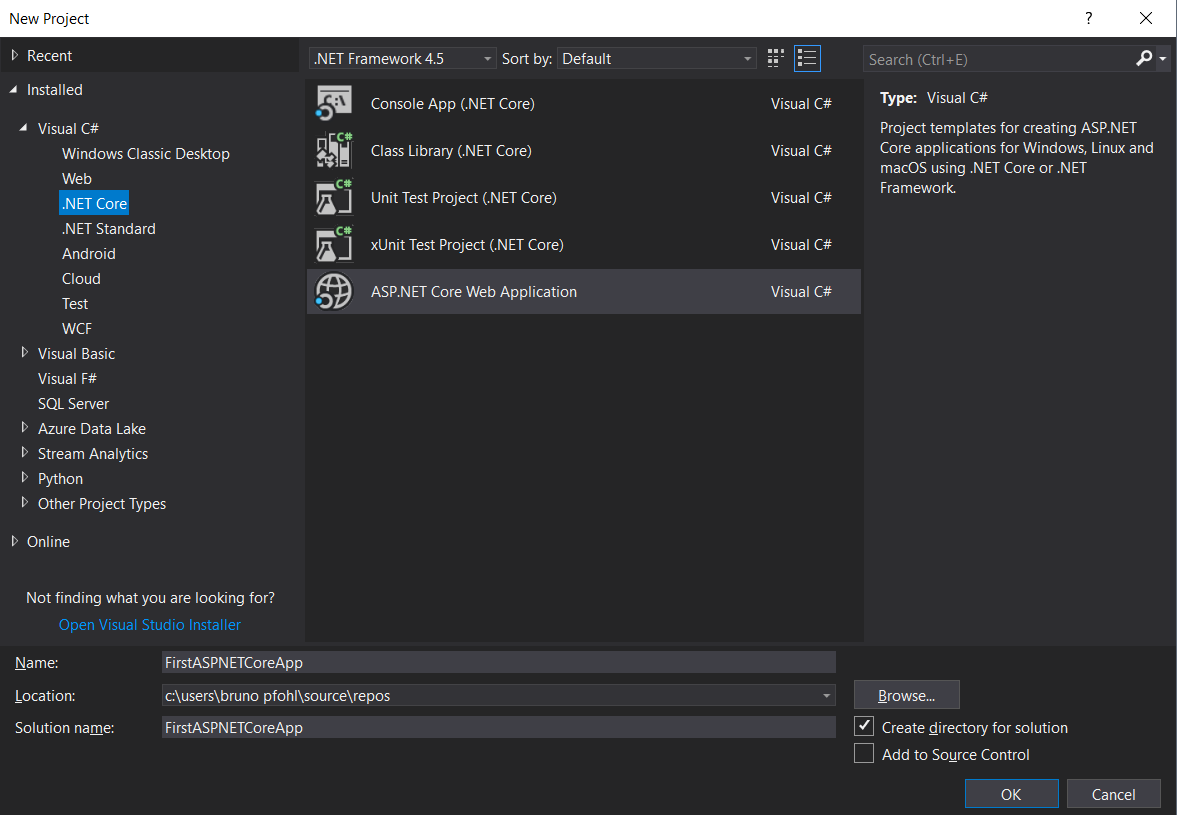
# Kapitola 4. První projekt

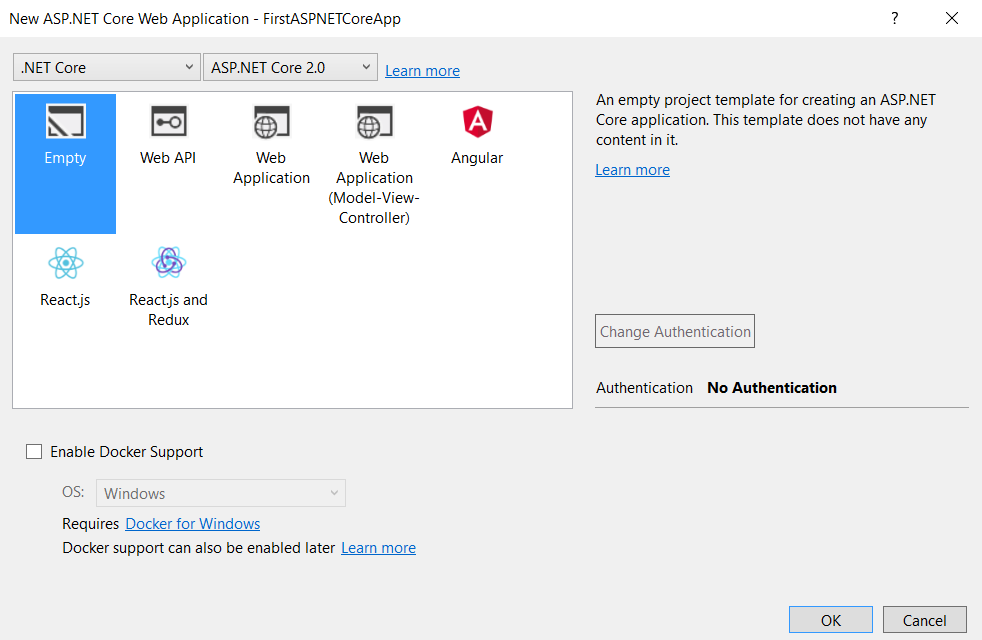
Na našem prvním projektu si pouze vysvětlíme základní strukturu ASP .NET Core projektu. Zatím tedy založíme pouze projekt bez implementace MVC architektury.

## Založení projektu

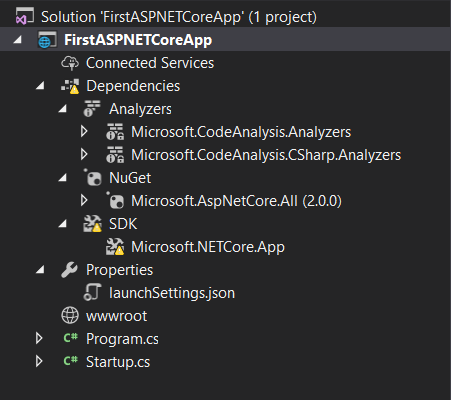
**Postup**

* Klikneme na založit nový projekt
* Zvolíme typ projektu „ASP. NET Core Web Application“
* Zvolíme šablonu „Empty“ a potvrdíme volbu





## Struktura projektu

Námi nově vytvořený projekt obsahuje pouze složku wwwroot a soubory Program.cs, Startup.cs a launchSettings.json.

### Obsah souboru Program.cs

Tento soubor by některým lidem, s předešlými zkušenostmi s programováním v jazyce C#, mohl připomínat konzolovou aplikaci. Je to pochopitelné, protože se o konzolovou aplikaci skutečně jedná. Tato aplikace slouží k nastavení a spuštění webového serveru.



**Metoda** **Main**

V této metodě dochází k sestavení a spuštění webového serveru. Pro sestavení webového serveru používá metodu BuildWebHost, které předává parametry args. Tyto parametry jí byly předány při spuštění aplikace. Pro spuštění serveru volá metodu Run.

**Metoda** **BuildWebHost**

Uvnitř této metody se vyskytuje kód, který slouží k vytvoření, nastavení a sestavení webového serveru. Na obrázku můžeme vidět, že se volá metoda CreateDefaultBuilder. Tato metoda vrací „builder“ s již přednastavenými vlastnostmi. Jednou z těchto vlastností je i to, že se jako webový server využívá **Kestrel**, který je multiplatformní a lze ho tedy spustit nejen na Windows, ale i na Mac Os a Linuxu.

Pomocí metody UseStartup specifikuje, kterou třídu má „builder“ využít k nastavení webového serveru. V našem případě se jedná o třídu Startup, která se nachází v souboru Startup.cs.

Obsah souboru Startup.cs

Třída Startup slouží k registrování služeb a sestavení pipeline. Jak můžete na obrázku vidět, třída nemá žádného předka. Správnou implementaci startup třídy tedy představuje jakákoliv třída, která dodržuje nutné konvence (tj. má metody ConfigureServices a Configure tak, jak je vidět na obrázku).

**Metoda** **ConfigureServices**

Obsahuje obslužný kód pro registraci a konfiguraci služeb, které aplikace potřebuje ke svojí funkčnosti. Právě v této metodě dochází k naplnění IoC Containeru (Inversion Of Control), kterému dal Microsoft název Service Provider. Service Provider se stará o předávání služeb (objektů) ostatním třídám, jejichž funkčnost je na těchto službách závislá. Tuto funkci frameworku si detailněji popíšeme v později.

**Metoda** **Configure**

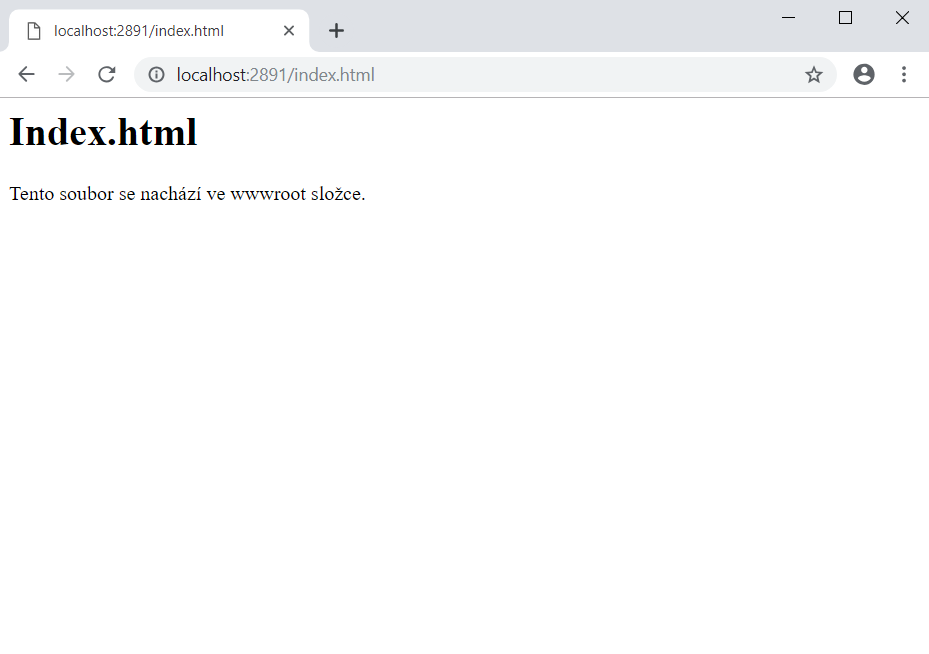
Stará se o sestavení request PIPELINE. Jinými slovy, určuje, jak bude webový server zpracovávat dotazy od uživatele. Ve většině případů budeme chtít předat dotaz nějakému middlewaru, který jej zpracuje a předá dalšímu middlewaru zaregistrovaném do PIPELINE. Kód této metody, který vidíte na obrázku pouze registruje middleware pro zobrazení stránky s detailní chybovou hláškou pro vývojáře. Pokud tedy v aplikaci při zpracování requestu nenastane žádná chyba, server vrátí pouze textový řetězec „Hello World!“, protože dotaz bude middlewarem předán dál.

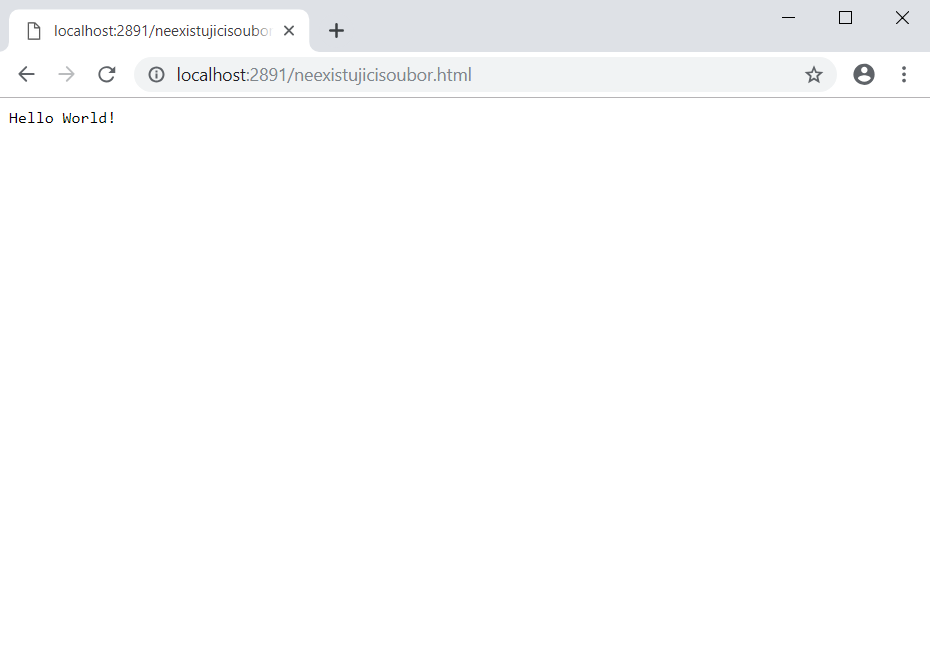
**Názorná ukázka funkce PIPELINE**



Do Configure metody ve třídě Startup bylo přidáno volání funkce UseStaticFiles, čímž byl zaregistrován middleware, který zajišťuje uživatelovi přístup k souborům ve složce wwwroot.

Volání této metody se nachází nad voláním app.Run, což v praxi znamená, že ke vrácení textového řetězce „Hello World“ dojde pouze tehdy, pokud nebude uživatelem vyžádán soubor nacházející se ve složce wwwroot nebo pokud nedojde k chybě. Důkazem toho, že se tak PIPELINE skutečně chová jsou následující dva obrázky.





Pro lepší pochopení doporučuji, abyste si zkusili zaměnit pořadí registrace middlewaru a otestovali, jaké změny to pro funkci aplikace přinese.